

# Machine Learning in Natural Language Processing

Fernando Pereira  
University of Pennsylvania

NASSLLI, June 2002

*Thanks to: William Bialek, John Lafferty, Andrew McCallum, Lillian Lee, Lawrence Saul, Yves Schabes, Stuart Shieber, Naftali Tishby*

## Introduction

ML in NLP

## Why ML in NLP

- Examples are easier to create than rules
- Rule writers miss low frequency cases
- Many factors involved in language interpretation
- People do it
  - AI
  - Cognitive science
- Let the computer do it
  - Moore's law
  - storage
  - lots of data

ML in NLP

## Classification

### ■ Document topic

#### Former Officials Say Enron Hid Gains During Crisis in California

By DAVID BARBOZA

In hopes of damping a political firestorm, former Enron officials said that the company kept as much as \$1.5 billion in profits off its books in late 2000 and early 2001.

politics, business

#### Era of the Big Fire Is Kindled at West's Doors

By TIMOTHY EGAN

A century-long policy of knocking down all fires has created fuel-filled forests which are likely to keep firefighters busier than ever.

national, environment

### ■ Word sense

*treasury bonds* □ *chemical bonds*

ML in NLP

VISIT...

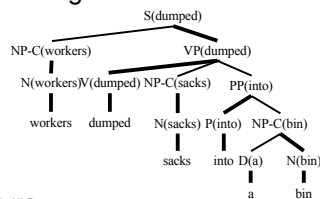
LANZAROTE  
*Caliente*.COM

## Analysis

### ■ Tagging

VBG      NNS      WDT      VBP      RP      NNS      JJ  
causing symptoms that show up decades later

### ■ Parsing



ML in NLP

## Language Modeling

### ■ Is this a likely English sentence?

$$\frac{P(\text{colorless green ideas sleep furiously})}{P(\text{furiously sleep ideas green colorless})} \approx 2 \times 10^5$$

### ■ Disambiguate noisy transcription

*It's easy to wreck a nice beach*  
*It's easy to recognize speech*

ML in NLP

## Inference

### ■ Translation

treasury bonds     $\square$     obrigações do tesouro  
covalent bonds     $\square$     ligações covalentes

### ■ Information extraction

acquirer  
acquired

#### Sara Lee to Buy 30% of DIM

Chicago, March 3 - Sara Lee Corp said it agreed to buy a 30 percent interest in Paris-based DIM S.A., a subsidiary of BIC S.A., at cost of about 20 million dollars. DIM S.A., a hosiery manufacturer, had sales of about 2 million dollars.

The investment includes the purchase of 5 million newly issued DIM shares valued at about 5 million dollars, and a loan of about 15 million dollars, it said. The loan is convertible into an additional 16 million DIM shares, it noted.

The proposed agreement is subject to approval by the French government, it said.

ML in NLP

## Machine Learning Approach

### ■ Algorithms that *write programs*

- Specify
  - Form of output programs
  - Accuracy criterion

### ■ *Input*: set of training examples

### ■ *Output*: program that performs as accurately as possible on the training examples

### ■ *But will it work on new examples?*

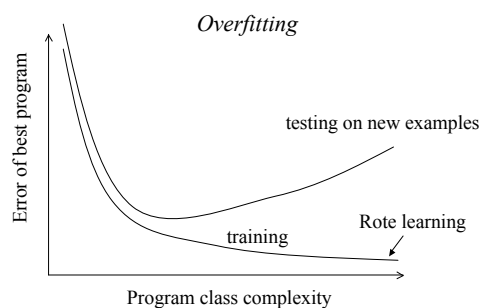
ML in NLP

## Fundamental Questions

- *Generalization*: is the learned program useful on new examples?
  - *Statistical learning theory*: quantifiable tradeoffs between number of examples, complexity of program class, and generalization error
- *Computational tractability*: can we find a good program quickly?
  - If not, can we find a good approximation?
- *Adaptation*: can the program learn quickly from new evidence?
  - Information-theoretic analysis: relationship between adaptation and compression

ML in NLP

## Learning Tradeoffs



ML in NLP

## Machine Learning Methods

- Classifiers
  - Document classification
  - Disambiguation disambiguation
- Structured models
  - Tagging
  - Parsing
  - Extraction
- Unsupervised learning
  - Generalization
  - Structure induction

ML in NLP

## Jargon

- *Instance*: event type of interest
  - Document and its class
  - Sentence and its analysis
  - ...
- *Supervised learning*: learn *classification function* from hand-labeled instances
- *Unsupervised learning*: exploit correlations to organize training instances
- *Generalization*: how well does it work on unseen data
- *Features*: map instance to set of elementary *events*

ML in NLP

## Classification Ideas

- Represent instances by *feature* vectors
  - Content
  - Context
- Learn function from feature vectors
  - Class
  - Class-probability distribution
- Redundancy is our friend: *many weak clues*

ML in NLP

## Structured Model Ideas

- *Interdependent* decisions
  - Successive parts-of-speech
  - Parsing/generation steps
  - Lexical choice
    - Parsing
    - Translation
- Combining decisions
  - Sequential decisions
  - Generative models
  - Constraint satisfaction

ML in NLP

## Unsupervised Learning Ideas

- *Clustering*: class induction
  - *Latent* variables
    - I'm thinking of sports □ more sporty words*
  - *Distributional* regularities
    - Know words by the company they keep
- Data compression
- Infer *dependencies among variables*: structure learning

ML in NLP

## Methodological Detour

- Empiricist/information-theoretic view:
  - words combine following their associations in previous material*
- Rationalist/generative view:
  - words combine according to a formal grammar in the class of possible natural-language grammars*

ML in NLP

## Chomsky's Challenge to Empiricism

- (1) *Colorless green ideas sleep furiously.*  
 (2) *Furiously sleep ideas green colorless.*
- ... It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticalness, these sentences will be ruled out on identical grounds as equally 'remote' from English. Yet (1), though nonsensical, is grammatical, while (2) is not.

Chomsky 57

ML in NLP

## The Return of Empiricism

- Empiricist methods *work*:
  - Markov models can capture a surprising fraction of the unpredictability in language
  - Statistical information retrieval methods beat alternatives
  - Statistical parsers are more accurate than competitors based on rationalist methods
  - Machine-learning, statistical techniques close to human performance in part-of-speech tagging, sense disambiguation
- *Just engineering tricks?*

ML in NLP

## Unseen Events

- Chomsky's implicit assumption: *any model must assign zero probability to unseen events*
  - naïve estimation of Markov model probabilities from frequencies
  - no *latent (hidden)* events
- Any such model *overfits* data: many events are likely to be missing in any finite sample
- The learned model *cannot generalize* to unseen data
- Support for *poverty of the stimulus* arguments

ML in NLP

## The Science of Modeling

- Probability estimates can be *smoothed* to accommodate unseen events
- *Redundancy* in language supports effective statistical inference procedures
- *the stimulus is richer than it might seem*
- *Statistical learning theory*: *generalization ability* of a model class can be measured independently of *model representation*
- *Beyond Markov models*: effects of *latent conditioning variables* can be estimated from data

ML in NLP

## Richness of the Stimulus

- Information *about*: *mutual information*
  - between linguistic and non-linguistic events
  - between parts of a linguistic event
- *Global coherence*:  
banks can now sell stocks and bonds
- *Word statistics* carry more information than it might seem
  - Markov models in speech recognition
  - Success of *bag-of-words* model in information retrieval
  - Statistical machine translation
- *How far* can these methods go?

ML in NLP

## Questions

- Generative or discriminative?
- Structured models: local classification or global constraint satisfaction?
- Does unsupervised learning help?

ML in NLP

## Classification

ML in NLP

## Generative or Discriminative?

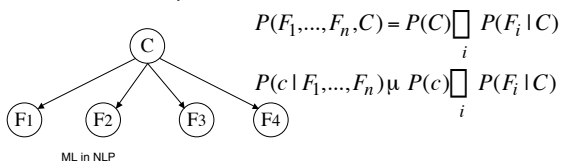
- Generative models
  - Estimate the instance-label distribution  
 $p(\mathbf{x}, y)$
- Discriminative models
  - Estimate the label-given-instance distribution  
 $p(y | \mathbf{x})$
  - Minimize an upper-bound on training error  

$$\sum_i \mathbb{I}[f(\mathbf{x}_i) \neq y_i]$$

ML in NLP

## Simple Generative Model

- Binary naïve Bayes:
  - Represent instances by sets of binary features
    - Does word occur in document
    - ...
  - Finite predefined set of classes



## Generative Claims

- Easy to train: just count
- *Language modeling*: probability of observed forms
- More robust
  - Small training sets
  - Label noise
- Full advantage of probabilistic methods

ML in NLP

## Discriminative Models

- Define functional form for  $p(y | \mathbf{x}; \theta)$
- Binary classification: define a discriminant function
 
$$y = \text{sign } h(\mathbf{x}; \theta)$$
- Adjust parameter(s)  $\theta$  to *maximize probability of training labels/minimize error*

ML in NLP

## Simple Discriminative Forms

- Linear discriminant function
 
$$h(\mathbf{x}; \theta_0, \theta_1, \dots, \theta_n) = \theta_0 + \sum_i \theta_i f_i(\mathbf{x})$$
- Logistic form:
 
$$P(+1 | \mathbf{x}) = \frac{1}{1 + \exp(-h(\mathbf{x}; \theta))}$$
- Multi-class exponential form (maxent):

$$h(\mathbf{x}, y; \theta_0, \theta_1, \dots, \theta_n) = \theta_0 + \sum_i \theta_i f_i(\mathbf{x}, y)$$

$$P(y | \mathbf{x}; \theta) = \frac{\exp h(\mathbf{x}, y; \theta)}{\sum_{y'} \exp h(\mathbf{x}, y'; \theta)}$$

ML in NLP

## Discriminative Claims

- Focus modeling resources on instance-to-label mapping
- Avoid restrictive probabilistic assumptions on instance distribution
- Optimize what you care about
- Higher accuracy

ML in NLP

## Classification Tasks

- Document categorization
  - News categorization
  - Message filtering
  - Web page selection
- Tagging
  - Named entity
  - Part-of-speech
  - Sense disambiguation
- Syntactic decisions
  - Attachment

ML in NLP

## Document Models

- Binary vector

$$f_t(\mathbf{d}) = t \in \mathbf{d}$$

- Frequency vector

$$\text{tf}(\mathbf{d}, t) = |\{t \in \mathbf{d}\}| \quad \text{idf}(\mathbf{d}, t) = |D| / |\{\mathbf{d} \in D : t \in \mathbf{d}\}|$$

raw frequency:  $r_t(\mathbf{d}) = \text{tf}(\mathbf{d}, t)$

$$\text{TF*IDF: } x_t(\mathbf{d}) = \log(1 + \text{tf}(\mathbf{d}, t)) \log(1 + \text{idf}(\mathbf{d}, t))$$

- N-gram language model

$$p(\mathbf{d} | c) = p(\mathbf{d} | c) \prod_{i=1}^{|\mathbf{d}|} p(d_i | d_1 \dots d_{i-1}; c)$$

$$p(d_i | d_1 \dots d_{i-1}; c) \approx p(d_i | d_{i-n} \dots d_{i-1}; c)$$

ML in NLP

## Term Weighting and Feature Selection

- Select or weigh most informative features
- TF\*IDF: adjust term weight by how document-specific the term is
- Feature selection:
  - Remove low, unreliable counts
  - Mutual information
  - Information gain
  - Other statistics

ML in NLP

## Documents vs. Vectors (1)

- Many documents have the same binary or frequency vector
- Document multiplicity must be handled correctly in probability models
- Binary naïve Bayes
 
$$p(\mathbf{f} | c) = \prod_t [f_t p(t | c) + (1 - f_t)(1 - p(t | c))]$$
- Multiplicity is not recoverable

ML in NLP

## Documents vs. Vectors (2)

- Document probability (unigram language model)

$$p(\mathbf{d} | c) = p(|\mathbf{d}| | c) \prod_{i=1}^{|\mathbf{d}|} p(d_i | c)$$

- Raw frequency vector probability

$$r_t = \text{tf}(\mathbf{d}, t)$$

$$p(\mathbf{r} | c) = p(L | c) L! \prod_t \frac{p(t | c)^{r_t}}{r_t!} \text{ where } L = \sum_t r_t$$

ML in NLP

## Documents vs. Vectors (3)

- Unigram model:

$$p(c | \mathbf{d}) = \frac{p(c) p(|\mathbf{d}| | c) \prod_{i=1}^{|\mathbf{d}|} p(d_i | c)}{\sum_c p(c) p(|\mathbf{d}| | c) \prod_{i=1}^{|\mathbf{d}|} p(d_i | c)}$$

- Vector model:

$$p(c | \mathbf{r}) = \frac{p(c) p(L | c) \prod_t p(t | c)^{r_t}}{\sum_c p(c) p(L | c) \prod_t p(t | c)^{r_t}}$$

ML in NLP

## Linear Classifiers

- Embedding into high-dimensional vector space
  - Geometric intuitions and techniques
  - Easier separability
    - Increase dimension with interaction terms
    - Nonlinear embeddings (kernels)
- Swiss Army knife

ML in NLP

## Kinds of Linear Classifiers

- Naïve Bayes
- Exponential models
- Large margin classifiers
  - Support vector machines (SVM)
  - Boosting
- Online methods
  - Perceptron
  - Winnow

ML in NLP

## Learning Linear Classifiers

- Rocchio

$$w_k = \max \left[ 0, \frac{\sum_{c \neq k} \sum_i x_i^c x_k^c}{|c|} - \frac{\sum_i x_i^k x_k^k}{|D| \sum_i |c|} \right]$$

- Widrow-Hoff

$$w \leftarrow w - 2\eta(w \cdot x_i - y_i)x_i$$

- (Balanced) winnow

$$y = \text{sign}(w^+ \cdot x - w^- \cdot x)$$

$$\text{positive error: } w^+ \leq \eta w^+, w^- \leq \eta w^-, \eta > 1 > \eta > 0$$

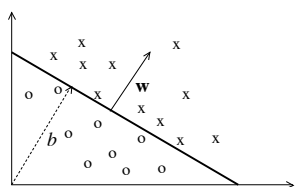
$$\text{negative error: } w^+ \leq \eta w^+, w^- \leq \eta w^-$$

ML in NLP

## Linear Classification

- Linear discriminant function

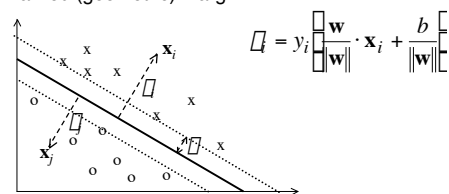
$$h(x) = w \cdot x + b = \sum_k w_k x_k + b$$



ML in NLP

## Margin

- Instance margin  $\Delta = y_i(w \cdot x_i + b)$
- Normalized (geometric) margin



- Training set margin  $\Delta$

ML in NLP

## Perceptron Algorithm

### ■ Given

- Linearly separable training set  $S$
- Learning rate  $\eta > 0$

$\mathbf{w} \leftarrow \mathbf{0}; b \leftarrow 0; R = \max_i \|\mathbf{x}_i\|$   
 repeat  
   for  $i = 1 \dots N$   
     if  $y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \leq 0$   
        $\mathbf{w} \leftarrow \mathbf{w} + \eta y_i \mathbf{x}_i$   
        $b \leftarrow b + \eta y_i R^2$   
 until there are no mistakes

ML in NLP

## Duality

- Final hypothesis is a linear combination of training points

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \quad \alpha_i \geq 0$$

### ■ Dual perceptron algorithm

$\alpha \leftarrow \mathbf{0}; b \leftarrow 0; R = \max_i \|\mathbf{x}_i\|$   
 repeat  
   for  $i = 1 \dots N$   
     if  $y_i \left( \sum_j \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b \right) \leq 0$   
        $\alpha_i \leftarrow \alpha_i + 1$   
        $b \leftarrow b + y_i R^2$   
 until there are no mistakes

ML in NLP

## Why Maximize the Margin?

- There is a constant  $c$  such that for any data distribution  $D$  with support in a ball of radius  $R$  and any training sample  $S$  of size  $N$  drawn from  $D$

$$p_{\text{err}}(h) \leq \frac{c}{N} \frac{R^2}{\gamma^2} \log^2 N + \log(1/\gamma) \geq 1 - \gamma$$

where  $\gamma$  is the margin of  $h$  in  $S$

ML in NLP

## Canonical Hyperplanes

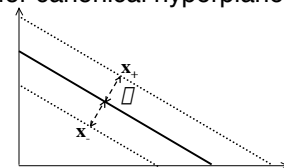
- Multiple representations for the same hyperplane

$$(\mathbf{w}, b) \quad \gamma > 0$$

- *Canonical hyperplane*: functional margin = 1
- Geometric margin for canonical hyperplane

$$\begin{aligned}
 \gamma &= \frac{1}{2} \frac{\mathbf{w} \cdot \mathbf{x}_+ - \mathbf{w} \cdot \mathbf{x}_-}{\|\mathbf{w}\|} \\
 &= \frac{1}{2\|\mathbf{w}\|} (\mathbf{w} \cdot \mathbf{x}_+ - \mathbf{w} \cdot \mathbf{x}_-) \\
 &= \frac{1}{\|\mathbf{w}\|}
 \end{aligned}$$

ML in NLP



## Convex Optimization (1)

- Constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \quad & f(\mathbf{w}) \\ \text{subject to} \quad & g_i(\mathbf{w}) \leq 0 \\ & h_j(\mathbf{w}) = 0 \end{aligned}$$

- Lagrangian function:

$$L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) = f(\mathbf{w}) + \sum_i \alpha_i g_i(\mathbf{w}) + \sum_j \beta_j h_j(\mathbf{w})$$

- Dual problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}, \boldsymbol{\beta}} \quad & \inf_{\mathbf{w}} L(\mathbf{w}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \\ \text{subject to} \quad & \alpha_i \geq 0 \end{aligned}$$

ML in NLP

## Convex Optimization (2)

- Kuhn-Tucker conditions:

- $f$  convex

- $g_i, h_j$  affine ( $h(\mathbf{w}) = \mathbf{A}\mathbf{w} - \mathbf{b}$ )

- Solution  $\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*$  must satisfy:

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \mathbf{w}} = 0$$

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\alpha}} = 0$$

$$\frac{\partial L(\mathbf{w}^*, \boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)}{\partial \boldsymbol{\beta}} = 0$$

$$\alpha_i^* g_i(\mathbf{w}^*) = 0$$

$$g_i(\mathbf{w}^*) \leq 0$$

$$\alpha_i^* \geq 0$$

Complementary condition:  
parameter is non-zero iff  
constraint is active

ML in NLP

## Maximizing the Margin (1)

- Given a separable training sample:

$$\min_{\mathbf{w}, b} \|\mathbf{w}\| = \mathbf{w} \cdot \mathbf{w} \text{ subject to } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$$

- Lagrangian:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \mathbf{w} \cdot \mathbf{w} + \sum_i \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1]$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} + \sum_i y_i \alpha_i \mathbf{x}_i = \mathbf{0}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_i y_i \alpha_i = 0$$

ML in NLP

## Maximizing the margin (2)

- Dual Lagrangian at stationary point:

$$W(\boldsymbol{\alpha}) = L(\mathbf{w}^*, b^*, \boldsymbol{\alpha}) = \sum_i \alpha_i \frac{1}{2} \sum_{i,j} y_i y_j \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j$$

- Dual maximization problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & W(\boldsymbol{\alpha}) \\ \text{subject to} \quad & \alpha_i \geq 0 \\ & \sum_i y_i \alpha_i = 0 \end{aligned}$$

- Maximum margin weight vector:

$$\mathbf{w}^* = \sum_i y_i \alpha_i^* \mathbf{x}_i \text{ with margin } \frac{1}{\|\mathbf{w}^*\|} = \left( \sum_{i \in \text{sv}} \alpha_i^* \right)^{1/2}$$

ML in NLP

## Building the Classifier

- Computing the offset (from primal constraints):

$$b^* = \frac{\max_{y_i = -1} \mathbf{w} \cdot \mathbf{x}_i + \min_{y_i = 1} \mathbf{w} \cdot \mathbf{x}_i}{2}$$

- Decision function:

$$h(\mathbf{x}) = \text{sgn}\left(\sum_i y_i \alpha_i \mathbf{x}_i \cdot \mathbf{x} + b^*\right)$$

ML in NLP

## Consequences

- *Complementarity* condition yields *support vectors*:

$$\alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1] = 0$$

$$\alpha_i > 0 \Rightarrow \mathbf{w} \cdot \mathbf{x}_i + b = y_i$$

- Functional margin of 1 implies minimum geometric margin

$$\gamma = 1 / \|\mathbf{w}\|$$

ML in NLP

## General SVM Form

- Margin maximization for an arbitrary kernel  $K$

$$\max_{\alpha} \quad \sum_i \alpha_i \quad \text{s.t.} \quad \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{subject to } \alpha_i \geq 0 \\ \sum_i y_i \alpha_i = 0$$

- Decision rule

$$h(\mathbf{x}) = \text{sgn}\left(\sum_i y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b^*\right)$$

ML in NLP

## Soft Margin

- Handles non-separable case
- Primal problem (2-norm):

$$\min_{\mathbf{w}, b, \xi} \quad \|\mathbf{w}\|^2 + C \sum_i \xi_i^2 \\ \text{subject to } y_i (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \\ \xi_i \geq 0$$

- Dual problem:

$$\max_{\alpha} \quad \sum_i \alpha_i \quad \text{s.t.} \quad \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j + \frac{1}{C} \sum_{ij} \alpha_i \alpha_j \\ \text{subject to } \alpha_i \geq 0 \\ \sum_i y_i \alpha_i = 0$$

ML in NLP

## Conditional Maxent Model

- Model form

$$p(y|\mathbf{x};\boldsymbol{\theta}) = \frac{\exp \sum_k \theta_k f_k(\mathbf{x}, y)}{Z(\mathbf{x};\boldsymbol{\theta})}$$

$$Z(\mathbf{x};\boldsymbol{\theta}) = \sum_y \exp \sum_k \theta_k f_k(\mathbf{x}, y)$$

- Useful properties

- Multi-class
- May use different features for different classes
- Training is convex optimization

ML in NLP

## Duality

- Maximize conditional log likelihood

$$\boldsymbol{\theta} = \arg \max_{\boldsymbol{\theta}} \sum_i \log p(y_i | \mathbf{x}_i; \boldsymbol{\theta})$$

- Maximizing conditional entropy

$$\tilde{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} p \left[ \sum_i \sum_y p(y | \mathbf{x}_i) \log p(y | \mathbf{x}_i) \right]$$

subject to constraints

$$\sum_i f_k(\mathbf{x}_i, y_i) = \sum_y p(y | \mathbf{x}_i) f_k(\mathbf{x}_i, y)$$

yields

$$\tilde{p}(y | \mathbf{x}) = p(y | \mathbf{x}; \tilde{\boldsymbol{\theta}})$$

ML in NLP

## Relationship to (Binary) Logistic Discrimination

$$p(+1 | \mathbf{x}) = \frac{\exp \sum_k \theta_k f_k(\mathbf{x}, +1)}{\exp \sum_k \theta_k f_k(\mathbf{x}, +1) + \exp \sum_k \theta_k f_k(\mathbf{x}, -1)}$$

$$= \frac{1}{1 + \exp \sum_k \theta_k (f_k(\mathbf{x}, +1) - f_k(\mathbf{x}, -1))}$$

$$= \frac{1}{1 + \exp \sum_k \theta_k g_k(\mathbf{x})}$$

ML in NLP

## Relationship to Linear Discrimination

- Decision rule

$$\text{sign} \left[ \log \frac{p(+1 | \mathbf{x})}{p(-1 | \mathbf{x})} \right] = \text{sign} \sum_k \theta_k g_k(\mathbf{x})$$

- Bias term: parameter for “always on” feature

- Question: relationship to other trainers for linear discriminant functions

ML in NLP

## Solution Techniques (1)

- Generalized iterative scaling (GIS)

- Parameter updates

$$\theta_k \leftarrow \theta_k + \frac{1}{C} \log \frac{\sum_i f_k(\mathbf{x}_i, y_i)}{\sum_i \sum_y p(y | \mathbf{x}_i; \theta) f_k(\mathbf{x}_i, y)}$$

- Requires that features add up to constant independent of instance or label (add *slack feature*)

$$\sum_k f_k(\mathbf{x}_i, y) = C \quad \forall i, y$$

ML in NLP

## Solution Techniques (2)

- Improved iterative scaling (IIS)

- Parameter updates

$$\theta_k \leftarrow \theta_k + \Delta_k$$

$$\sum_i f_k(\mathbf{x}_i, y_i) = \sum_i \sum_y p(y | \mathbf{x}_i; \theta) f_k(\mathbf{x}_i, y) e^{\Delta_k f^\#(\mathbf{x}_i, y)}$$

$$f^\#(\mathbf{x}, y) = \sum_k f_k(\mathbf{x}, y)$$

- For binary features reduces to solving a polynomial with positive coefficients
- Reduces to GIS if feature sum constant

ML in NLP

## Deriving IIS (1)

- Conditional log-likelihood

$$l(\theta) = \sum_i \log p(y_i | \mathbf{x}_i; \theta)$$

- Log-likelihood update

$$\begin{aligned} l(\theta + \Delta) - l(\theta) &= \sum_i \Delta \cdot f(\mathbf{x}_i, y_i) + \log \frac{Z(\mathbf{x}_i; \theta + \Delta)}{Z(\mathbf{x}_i; \theta)} \\ &= \sum_i \Delta \cdot f(\mathbf{x}_i, y_i) + \sum_i \log \sum_y \frac{e^{(\theta + \Delta) \cdot f(\mathbf{x}_i, y)}}{Z(\mathbf{x}_i; \theta)} \\ &= \sum_i \Delta \cdot f(\mathbf{x}_i, y_i) + \sum_i \log \sum_y p(y | \mathbf{x}_i; \theta) e^{\Delta \cdot f(\mathbf{x}_i, y)} \\ (\log x - \log x) &\geq \underbrace{\sum_i \Delta \cdot f(\mathbf{x}_i, y_i) + N \sum_i \sum_y p(y | \mathbf{x}_i; \theta) e^{\Delta \cdot f(\mathbf{x}_i, y)}}_{A(\Delta)} \end{aligned}$$

ML in NLP

## Deriving IIS (2)

- By Jensen's inequality:

$$A(\Delta) \geq \sum_i \Delta \cdot f(\mathbf{x}_i, y_i) + N \Delta$$

$$\sum_i \sum_y p(y | \mathbf{x}_i; \theta) \sum_k \frac{f_k(\mathbf{x}_i, y)}{f^\#(\mathbf{x}_i, y)} e^{\Delta_k f^\#(\mathbf{x}_i, y)} = B(\Delta)$$

$$\frac{\partial B(\Delta)}{\partial \Delta_k} = \sum_i f_k(\mathbf{x}_i, y_i) + \sum_i \sum_y p(y | \mathbf{x}_i; \theta) f_k(\mathbf{x}_i, y) e^{\Delta_k f^\#(\mathbf{x}_i, y)}$$

- Maximize lower bound on update

ML in NLP

### Solution Techniques (3)

- GIS very slow if slack variable takes large values
- IIS faster, but still problematic
- Recent suggestion: use standard convex optimization techniques
  - Eg. Conjugate gradient
  - Some evidence of faster convergence

ML in NLP

### Gaussian Prior

- Log-likelihood gradient

$$\frac{\partial l(\beta)}{\partial \beta_k} = \sum_i f_k(\mathbf{x}_i, y_i) \sum_y p(y | \mathbf{x}_i; \beta) f_k(\mathbf{x}_i, y) \beta_k$$

- Modified IIS update

$$\beta_k \leftarrow \beta_k + \beta_k$$

$$\sum_i f_k(\mathbf{x}_i, y_i) =$$

$$\sum_i \sum_y p(y | \mathbf{x}_i; \beta) f_k(\mathbf{x}_i, y) e^{\beta_k f^{\#}(\mathbf{x}_i, y)} + \frac{\beta_k + \beta_k}{\beta_k^2}$$

$$f^{\#}(\mathbf{x}, y) = \sum_k f_k(\mathbf{x}, y)$$

ML in NLP

### Instance Representation

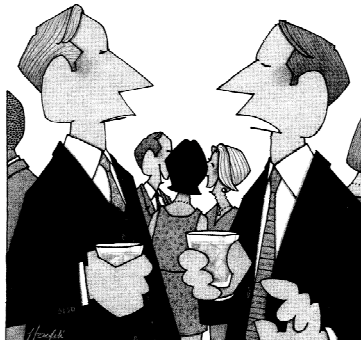
- Fixed-size instance (PP attachment): binary features
  - Word identity
  - Word class
- Variable-size instance (document classification)
  - Word identity
  - Word relative frequency in document

ML in NLP

### Enriching Features

- Word  $n$ -grams
- Sparse word  $n$ -grams
- Character  $n$ -grams (noisy transcriptions: speech, OCR)
- Unknown word features: suffixes, capitalization
- Feature combinations (cf.  $n$ -grams)

ML in NLP



I understood each and every word you said but not the order in which they appeared.

ML in NLP

## Structured Models: Finite State

ML in NLP

## Structured Model Applications

- Language modeling
- Story segmentation
- POS tagging
- Information extraction (IE)
- (Shallow) parsing

ML in NLP

## Structured Models

- Assign a labeling to a sequence
  - Story segmentation
  - POS tagging
  - Named entity extraction
  - (Shallow) parsing

ML in NLP

## Constraint Satisfaction in Structured Models

- Train to minimize labeling loss

$$\hat{y} = \arg \min_{\hat{y}} \sum_i \text{Loss}(\mathbf{x}_i, \mathbf{y}_i \mid \hat{y})$$

- Computing the best labeling:  

$$\arg \min_{\mathbf{y}} \text{Loss}(\mathbf{x}, \mathbf{y} \mid \hat{y})$$
- Efficient minimization requires:
  - A common currency for local labeling decisions
  - Efficient algorithm to combine the decisions

ML in NLP

## Local Classification Models

- Train to minimize the per-decision loss in context

$$\hat{y} = \arg \min_{\hat{y}} \sum_i \sum_{0 \leq j < |\mathbf{x}_i|} \text{loss}(\mathbf{y}_{i,j} \mid \mathbf{x}_i, \mathbf{y}_i^{(j)}; \hat{y})$$

- Apply by guessing context and finding each lowest-loss label:

$$\arg \min_{y_j} \text{loss}(y_j \mid \mathbf{x}, \hat{\mathbf{y}}^{(j)}; \hat{y})$$

ML in NLP

## Structured Model Claims

- Constraint satisfaction
  - Principled
  - Probabilistic interpretation allows model composition
  - Efficient optimal decoding
- Local classification
  - Wider range of models
  - More efficient training
  - Heuristic decoding comparable to pruning in global models

ML in NLP

## Example: Language Modeling

- $n$ -gram ( $n-1$  order Markov) model:

$$w_1 \cdots \overbrace{[w_{k-n+1} \cdots w_{k-1} w_k]}^n \cdots$$

$$P(w_k \mid w_1 \cdots w_{k-1}) \approx P(w_k \mid w_{k-n+1} \cdots w_{k-1})$$

- example, character  $n$ -grams (Shannon, Lucky):

$n$

0	XFOML RXXHRJFFJUU ZLPWCFWKCYJ
1	OCRO HLI RGWR NMIELWIS EU LL
2	ON IE ANTSOUTINYS ARE T INCTORE ST
3	IN NO IST LAT WHEY CRATICT FROURE
4	ABOVERY UPONDULTS WELL THE CODERST

ML in NLP

## Markov's Unreasonable Effectiveness

- Entropy estimates for English
 

model	bits/char
compress	4.43
<b>word trigrams</b> (Brown <i>et al</i> 92)	<b>1.75</b>
human prediction (Cover & King 78)	1.34
- Local word relations dominate statistics (Jelinek):
 

1	The	are	to	know	the	issues	necessary	role
2	This	will	have	this	problems	data	thing	
2	One	the	understand	these	<b>the</b>	information	that	
7	Please	<b>need</b>	use	problem		people	point	
9	<b>We</b>	insert	<b>all</b>			tools	<b>issues</b>	
98		<b>resolve</b>				old		
1641						<b>important</b>		

ML in NLP

## Limits of Markov Models

- No dependency
- Likelihoods based on sequencing, not dependency

1 The are to know the issues necessary role  
 2 This will have this problems data thing  
 2 One the understand these **the** information that  
 7 Please **need** use problem people point  
 9 **We** insert **all** tools **issues**  
 98 **resolve** old  
 1641 **important**

ML in NLP

## Unseen Events (1)

- What's the probability of unseen events?
- Bias* forces nonzero probabilities for some unseen events
- Typical bias: tie probabilities of *related* events
  - specific unseen event  $\square$  general seen event  
 $\text{eat pineapple} \square \text{eat } \_$
  - event decomposition*: event  $\square$  event<sub>1</sub>  $\square$  event<sub>2</sub>  
 $\text{eat pineapple} \square \text{eat } \_ \square \text{pineapple}$
  - Factoring via latent variables*:

$$P(\text{eat} \mid \text{pineapple}) \square \square P(\text{eat} \mid C) \square P(C \mid \text{pineapple})$$

ML in NLP

## Unseen Events (2)

- Discount* estimates for seen events
- Use leftover for unseen events
- How to allocate leftover?
  - Back-off* from unseen event to less specific seen events:  $n$ -gram to  $n-1$ -gram
  - Hypothesize hidden cause for unseen events: *latent variable* model
  - Relate unseen event to *distributionally similar seen events*

ML in NLP

## Important Detour: Latent Variable Models

ML in NLP

## Expectation-Maximization (EM)

### ■ Latent (hidden) variable models

$$p(y, \mathbf{x}, \mathbf{z} | \boldsymbol{\theta})$$

$$p(y, \mathbf{x} | \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(y, \mathbf{x}, \mathbf{z} | \boldsymbol{\theta})$$

### ■ Examples:

- Mixture models
- Class-based models (hidden classes)
- Hidden Markov models

ML in NLP

## Maximizing Likelihood

### ■ Data log-likelihood

$$D = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$$

$$L(D | \boldsymbol{\theta}) = \sum_i \log p(\mathbf{x}_i, y_i) = \sum_{\mathbf{x}, y} \tilde{p}(\mathbf{x}, y) \log p(\mathbf{x}, y | \boldsymbol{\theta})$$

$$\tilde{p}(\mathbf{x}, y) = \frac{|\{i : \mathbf{x}_i = \mathbf{x}, y_i = y\}|}{N}$$

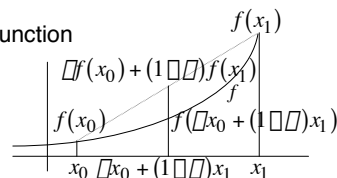
### ■ Find parameters that maximize (log-)likelihood

$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \sum_{\mathbf{x}, y} \tilde{p}(\mathbf{x}, y) \log p(\mathbf{x}, y | \boldsymbol{\theta})$$

ML in NLP

## Convenient Lower Bounds (1)

### ■ Convex function

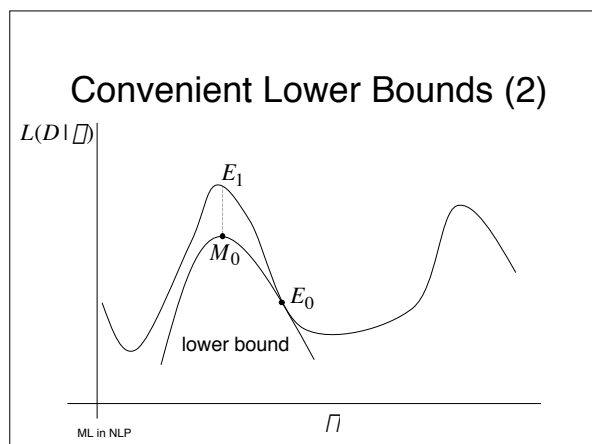


### ■ Jensen's inequality

$$f\left(\sum_x p(x)x\right) \geq \sum_x p(x)f(x)$$

if  $f$  is *convex* and  $p$  is a probability density

ML in NLP



### Auxiliary Function

- Find a convenient non-negative function that lower-bounds likelihood increase

$$L(D|\theta) - L(D|\theta_0) \geq Q(\theta|\theta_0) \geq 0$$

- Maximize lower bound:

$$\theta_{i+1} = \arg \max_{\theta} Q(\theta|\theta_i)$$

ML in NLP

### Comments

- Likelihood keeps increasing, but
  - Can get stuck in local maximum (or saddle point!)
  - Can oscillate between different local maxima with same log-likelihood
- If maximizing auxiliary function is too hard, find some  $\theta$  that increases likelihood: generalized EM (GEM)
- Sum over hidden variable values can be exponential if not done carefully (sometimes not possible)

ML in NLP

### Example: Mixture Model

- Base distributions

$$p_i(y) : 1 \leq i \leq m$$

- Mixture coefficients

$$\alpha_i \geq 0 \quad \sum_{i=1}^m \alpha_i = 1$$

- Mixture distribution

$$p(y|\theta) = \sum_i \alpha_i p_i(y)$$

ML in NLP

## Auxiliary Quantities

- Mixture coefficient  $i$  = prior probability of being in class  $I$
- Joint probability

$$p(c, y | \mathbf{x}) = \sum_c p_c(y) p(c | \mathbf{x})$$

- Auxiliary function

$$Q(\mathbf{c} | \mathbf{x}) = \sum_y \tilde{p}(y) \sum_c p(c | y, \mathbf{x}) \log \frac{p(y, c | \mathbf{x})}{p(y, c | \mathbf{x})}$$

ML in NLP

## Solution

- E step:

$$C_i = \frac{1}{\sum_j C_j} \sum_y \tilde{p}(y) \frac{C_i p_i(y)}{\sum_j C_j p_j(y)}$$

- M-step:

$$C_i = \frac{C_i}{\sum_j C_j}$$

ML in NLP

## More Finite-State Models

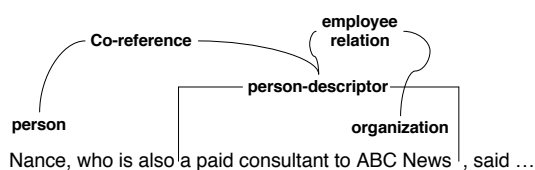
ML in NLP

## Example: Information Extraction

- *Given*: types of *entities* and *relationships* we are interested in
  - People, places, organizations, dates, amounts, materials, processes, ...
  - Employed by, located in, used for, arrived when, ...
- *Find all* entities and relationships of the given types in source material
- *Collect* in suitable database

ML in NLP

## IE Example



- Rely on:
  - Syntactic structure
  - Phrase classification

ML in NLP

## IE Methods

- *Partial matching*:
  - Hand-built patterns
  - Automatically-trained hidden Markov models
  - Cascaded finite-state transducers
- *Parsing-based*:
  - *Parse* the whole text:
    - Shallow parser (chunking)
    - Automatically-induced grammar
  - *Classify* phrases and phrase relations as desired entities and relationships

ML in NLP

## Global Constraint Models

- Train to minimize labeling loss
 
$$\hat{\mathcal{D}} = \arg \min_{\mathcal{D}} \sum_i \text{Loss}(\mathbf{x}_i, \mathbf{y}_i \mid \mathcal{D})$$
- Computing the best labeling:
 
$$\arg \min_{\mathbf{y}} \text{Loss}(\mathbf{x}, \mathbf{y} \mid \hat{\mathcal{D}})$$
- Efficient minimization requires:
  - A common currency for local labeling decisions
  - A dynamic programming algorithm to combine the decisions

ML in NLP

## Local Classification Models

- Train to minimize the per-symbol loss in context
 
$$\hat{\mathcal{D}} = \arg \min_{\mathcal{D}} \sum_i \sum_{0 \leq j < |\mathbf{x}_i|} \text{loss}(y_{i,j} \mid \mathbf{x}_i, \mathbf{y}_i^{(j)}; \mathcal{D})$$
- Apply by guessing context and finding each lowest-loss label:
 
$$\arg \min_{y_j} \text{loss}(y_j \mid \mathbf{x}, \hat{\mathbf{y}}^{(j)}; \hat{\mathcal{D}})$$

ML in NLP

## Structured Model Claims

- Global constraint
  - Principled
  - Probabilistic interpretation allows model composition
  - Efficient optimal decoding
- Local classifier
  - Wider range of models
  - More efficient training
  - Heuristic decoding comparable to pruning in global models

ML in NLP

## Generative vs. Discriminative

- Hidden Markov models (HMMs):  
generative, global
- Conditional exponential models (MEMMs, CRFs): discriminative, global
- Boosting, winnow: discriminative, local

ML in NLP

## Generative Models

- Stochastic process that generates instance-label pairs
  - Process structure
  - Process parameters
- (Hypothesize structure)
- Estimate parameters from training data

ML in NLP

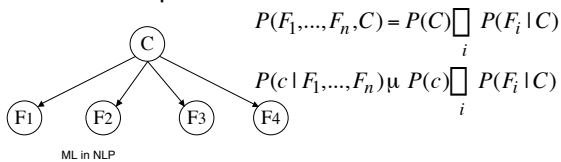
## Model Structure

- Decompose the generation of instances into elementary steps
- Define dependencies between steps
- Parameterize the dependencies
- Useful descriptive language: *graphical models*

ML in NLP

## Binary Naïve Bayes

- Represent instances by sets of binary features
  - Does word occur in document
  - ...
- Finite predefined set of classes

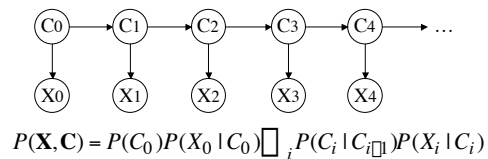


$$P(F_1, \dots, F_n, C) = P(C) \prod_i P(F_i | C)$$

$$P(c | F_1, \dots, F_n) \propto P(c) \prod_i P(F_i | C)$$

## Discrete Hidden Markov Model

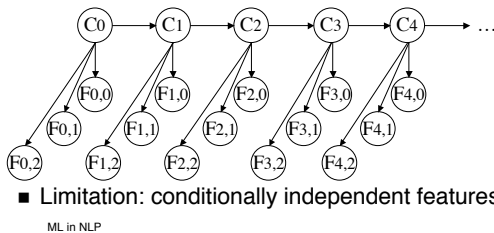
- Instances: symbol sequences
- Labels: class sequences



$$P(\mathbf{X}, \mathbf{C}) = P(C_0)P(X_0 | C_0) \prod_i P(C_i | C_{i-1})P(X_i | C_i)$$

## Generating Multiple Features

- Instances: sequences of feature sets
  - Word identity
  - Word properties (eg. spelling, capitalization)
- Labels: class sequences



- Limitation: conditionally independent features

## Independence or Intractability

- Trees are good: each node has a single immediate ancestor, joint probability computed in linear time
- But that forces features to be conditionally independent given the class
- Unrealistic
  - Suffixes and capitalization
  - "San" and "Francisco" in document

ML in NLP

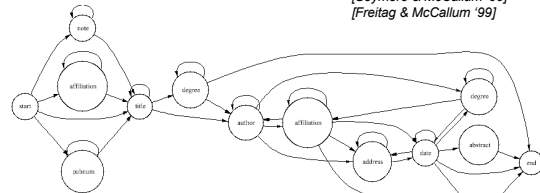
## Score Card

- ✓ No independence assumptions
- ✓ Richer features: combinations of existing features
- Optimization problem for parameters
- Limited probabilistic interpretation
- Insensitive to input distribution

ML in NLP

## Information Extraction with HMMs

[Seymore & McCallum '99]  
[Freitag & McCallum '99]



- Parameters =  $P(s|s')$ ,  $P(o|s)$  for all states in  $S = \{s_1, s_2, \dots\}$
- Observations: words
- Training: maximize probability of observations (+ prior).
- For IE, states indicate "database field".

ML in NLP

## Problems with HMMs

1. Would prefer richer representation of text: multiple *overlapping* features, whole chunks of text
  - Example word features:
    - identity of word
    - word is in all caps
    - word ends in "-tion"
    - word is part of a noun phrase
    - word is in bold font
    - word is on left hand side of page
    - word is under node X in WordNet
  - Example line features:
    - length of line
    - line is centered
    - percent of non-alphabets
    - total amount of white space
    - line contains two verbs
    - line begins with a number
    - line is grammatically a question
2. HMMs are generative models. Generative models do not handle easily overlapping, non-independent features. Would prefer a *conditional* model:  $P(\{s \dots\} | \{o \dots\})$ .

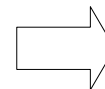
ML in NLP

## Solution: Conditional Model

**Hidden Markov model**

$$P(s|s')$$

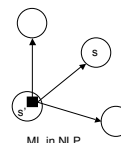
$$P(o|s)$$



**Maximum entropy Markov model**

$$P(s|o, s')$$

(represented by exponential model)

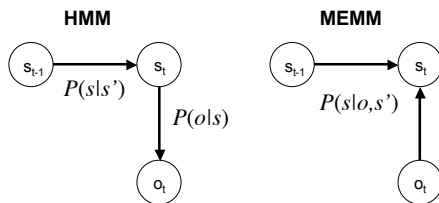


ML in NLP

For the time being, capture dependency on  $s'$  with  $|S|$  independent functions,  $P_s(s|o)$

Each state contains a "next-state classifier" black box, that, given the next observation, will produce a probability distribution over possible next states,  $P_s(s|o)$ .

## Two Sequence Models



- Standard belief propagation: forward-backward procedure.
- Viterbi and Baum-Welch follow naturally.

ML in NLP

## Transition Features

- Model  $P_{s'}(s|o)$  in terms of multiple arbitrary *overlapping* (binary) features.
- Example observation predicates:
  - $o$  is the word "apple"
  - $o$  is capitalized
  - $o$  is on a left-justified line
- Feature  $f$  depends on both a predicate  $b$
- ... and a destination state  $s$ .

$$f_{\langle b, s \rangle}(o, s') = \begin{cases} 1 & \text{if } b(o) \text{ is true and } s' = s \\ 0 & \text{otherwise} \end{cases}$$

ML in NLP

## Next-State Classifier

- Per-state conditional maxent model

$$P_{s'}(s|o) = \frac{1}{Z(o, s')} \exp \left( \sum_{\langle b, q \rangle} \left[ \sum_{\langle b, q \rangle} f_{\langle b, q \rangle}(o, s) \right] \right)$$

- Training: each state model *independently* from labeled sequences

ML in NLP

## Example: Q-A pairs from FAQ

X-NNTP-Poster: NewsHound v1.33  
Archive-name: acorn/faq/part2  
Frequency: monthly

2.6) What configuration of serial cable should I use?

Here follows a diagram of the necessary connections for common terminal programs to work properly. They are as far as I know the informal standard agreed upon by commercial comms software developers for the Arc.

Pins 1, 4, and 8 must be connected together inside the 9 pin plug. This is to avoid the well known serial port chip bugs. The modem's DCD (Data Carrier Detect) signal has been re-routed to the Arc's RI (Ring Indicator) most modems broadcast a software RING signal anyway, and even then it's really necessary to detect it for the model to answer the call.

2.7) The sound from the speaker port seems quite muffled.  
How can I get unfiltered sound from an Acorn machine?

All Acorn machine are equipped with a sound filter designed to remove high frequency harmonics from the sound output. To bypass the filter, hook into the Unfiltered port. You need to have a capacitor. Look for LM324 (chip 39) and hook the capacitor like this:

ML in NLP

## Experimental Data

### ■ 38 files belonging to 7 UseNet FAQs

```
<head>      X-NNTP-Poster: NewsHound v1.33
<head>      Archive-name: acorn/faq/part2
<head>      Frequency: monthly
<head>
<question>  2.6) What configuration of serial cable should I use?
<answer>
<answer>      Here follows a diagram of the necessary connection
<answer>      programs to work properly.  They are as far as I know
<answer>      agreed upon by commercial comms software developers fo
<answer>
<answer>      Pins 1, 4, and 8 must be connected together inside
<answer>      is to avoid the well known serial port chip bugs.  The
```

### ■ Procedure: For each FAQ, train on one file, test on other; average.

ML in NLP

## Features in Experiments

begins-with-number	contains-question-mark
begins-with-ordinal	contains-question-word
begins-with-punctuation	ends-with-question-mark
begins-with-question-word	first-alpha-is-capitalized
begins-with-subject	indented
blank	indented-1-to-4
contains-alphanum	indented-5-to-10
contains-bracketed-number	more-than-one-third-space
contains-http	only-punctuation
contains-non-space	prev-is-blank
contains-number	prev-begins-with-ordinal
contains-pipe	shorter-than-30

ML in NLP

## Models Tested

- **ME-Stateless**: A single maximum entropy classifier applied to each line independently.
- **TokenHMM**: A fully-connected HMM with four states, one for each of the line categories, each of which generates individual tokens (groups of alphanumeric characters and individual punctuation characters).
- **FeatureHMM**: Identical to TokenHMM, only the lines in a document are first converted to sequences of features.
- **MEMM**: maximum entropy Markov model

ML in NLP

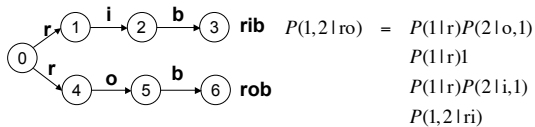
## Results

<i>Learner</i>	<i>Segmentation precision</i>	<i>Segmentation recall</i>
<b>ME-Stateless</b>	0.038	0.362
<b>TokenHMM</b>	0.276	0.140
<b>FeatureHMM</b>	0.413	0.529
<b>MEMM</b>	0.867	0.681

ML in NLP

## Label Bias Problem

- Example (after Bottou '91):



- Bias toward states with fewer outgoing transitions.
- Per-state normalization does not allow the required score  $(1,2|ro) \ll \text{score}(1,2|ri)$ .

ML in NLP

## Proposed Solutions

- *Determinization*:

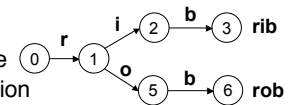
- not always possible
- state-space explosion

- *Fully-connected models*:

- lacks prior structural knowledge.

- *Conditional random fields (CRFs)*:

- Allow some transitions to *vote* more strongly than others
- *Whole sequence* rather than per-state normalization



ML in NLP

## From HMMs to CRFs

$$\begin{aligned}
 \mathbf{s} &= s_1 \dots s_n & \mathbf{o} &= o_1 \dots o_n \\
 \text{HMM} \quad P(\mathbf{s}|\mathbf{o}) &\propto \prod_{t=1}^n P(s_t | s_{t-1}) P(o_t | s_t) \\
 \text{MEMM} \quad P(\mathbf{s}|\mathbf{o}) &= \prod_{t=1}^n P(s_t | s_{t-1}, o_t) \\
 &= \prod_{t=1}^n \frac{1}{Z(s_{t-1}, o_t)} \exp \left( \sum_f \phi_f(s_t, s_{t-1}) + \sum_g \phi_g(s_t, o_t) \right) \\
 \text{CRF} \quad P(\mathbf{s}|\mathbf{o}) &= \frac{1}{Z(\mathbf{o})} \prod_{t=1}^n \exp \left( \sum_f \phi_f(s_t, s_{t-1}) + \sum_g \phi_g(s_t, o_t) \right)
 \end{aligned}$$

ML in NLP

## CRF General Form

- State sequence is a Markov random field conditioned on the observation sequence.

- Model form:  $P(\mathbf{s}|\mathbf{o}) = \frac{1}{Z(\mathbf{o})} \exp \left( \sum_{t=1}^n \sum_f \phi_f(s_{t-1}, s_t, \mathbf{o}, t) \right)$

- Features represent the dependency between successive states conditioned on the observations

- Dependence on *whole observation sequence*  $\mathbf{o}$  (not possible in HMMs).

ML in NLP

## Efficient Estimation

### ■ Matrix notation

$$M_t(s', s | \mathbf{o}) = \exp \left[ \phi_t(s', s | \mathbf{o}) \right]$$

$$\phi_t(s', s | \mathbf{o}) = \sum_f \sum_{i'} f(s_{i'}, s_t, \mathbf{o}, i')$$

$$P_{\square}(\mathbf{s} | \mathbf{o}) = \frac{1}{Z_{\square}(\mathbf{o})} \prod_t M_t(s_{i'}, s_t | \mathbf{o})$$

$$Z_{\square}(\mathbf{o}) = (M_1(\mathbf{o}) M_2(\mathbf{o}) \cdots M_{n+1}(\mathbf{o}))_{\text{start, stop}}$$

### ■ Efficient normalization: *forward-backward* algorithm

ML in NLP

## Forward-Backward Calculations

### ■ For any *path function* $G(s) = \sum_i g_t(s_{i'}, s_t)$

$$\begin{aligned} E_{\square} G &= \sum_s P_{\square}(\mathbf{s} | \mathbf{o}) G(\mathbf{s}) \\ &= \sum_{t, s', s} \frac{\bar{\phi}_t(s' | \mathbf{o}) g_{t+1}(s', s) M_{t+1}(s', s | \mathbf{o}) \bar{\phi}_{t+1}(s | \mathbf{o})}{Z_{\square}(\mathbf{o})} \end{aligned}$$

$$\bar{\phi}_t(\mathbf{o}) = \bar{\phi}_{t-1}(\mathbf{o}) M_t(\mathbf{o})$$

$$\bar{\phi}_t(\mathbf{o}) \square = M_{t+1}(\mathbf{o}) \bar{\phi}_{t+1}(\mathbf{o})$$

$$Z_{\square}(\mathbf{o}) = \bar{\phi}_{n+1}(\text{end} | \mathbf{o}) = \bar{\phi}_0(\text{start} | \mathbf{o})$$

ML in NLP

## Training

### ■ Maximize $L(\square) = \sum_k \log P_{\square}(\mathbf{s}_k | \mathbf{o}_k)$

### ■ Log-likelihood *gradient*

$$\frac{\partial L(\square)}{\partial \phi_f} = \sum_k \#_f(\mathbf{s}_k | \mathbf{o}_k) \square \sum_k E_{\square} \#_f(\mathbf{S} | \mathbf{o}_k)$$

$$\#_f(\mathbf{s} | \mathbf{o}) = \sum_t f(s_{i'}, s_t, \mathbf{o}, t)$$

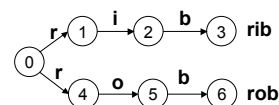
### ■ Methods: iterative scaling, *conjugate gradient*

### ■ Comparable to standard Baum-Welch

ML in NLP

## Label Bias Experiment

### ■ Data source: noisy version of



### ■ $P(\text{intended symbol}) = 29/30$ , $P(\text{other}) = 1/30$ .

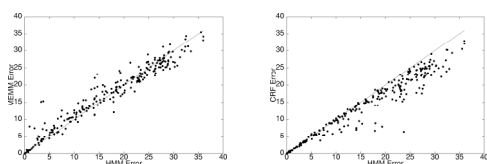
### ■ Train both an MEMM and a CRF with identical topologies on data from the source.

### ■ Compute decoding error: CRF 4.6%, MEMM 42% (2,000 training samples, 500 test)

ML in NLP

## Mixed-Order Sources

- Data generated by mixing sparse first and second order HMMs with varying mixing coefficient.
- Modeled by first-order HMM, MEMM and CRF (without contextual or overlapping features).



ML in NLP

## Part-of-Speech Tagging

- Trained on 50% of the 1.1 million words in the Penn treebank. In this set, 5.45% of the words occur only once, and were mapped to "oov".
- Experiments with two different sets of features:
  - traditional: just the words
  - take advantage of power of conditional models: use words, plus overlapping features: capitalized, begins with #, contains hyphen, ends in -ing, -ogy, -ed, -s, -ly, -ion, -tion, -ity, -ies.

ML in NLP

## POS Tagging Results

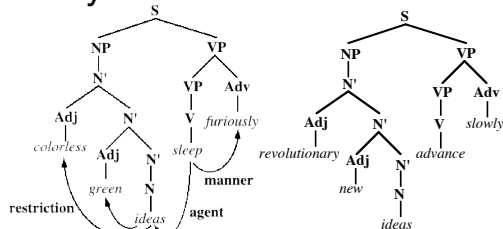
<i>model</i>	<i>error</i>	<i>oov error</i>
HMM	5.69%	45.99%
MEMM	6.37%	54.61%
CRF	5.55%	48.05%
MEMM+	4.81%	26.99%
CRF+	4.27%	23.76%

ML in NLP

## Structured Models: Stochastic Grammars

ML in NLP

## Beyond Finite State



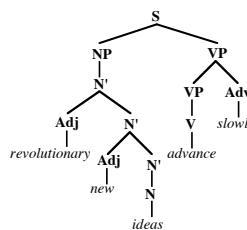
- Constituency and meaningful relations  
*What sleeps? How does it sleep? What kind of ideas?*

- Related sentences

*Sleep furiously is all that colorless green ideas do.*

ML in NLP

## Stochastic Context-Free Grammars



$S \rightarrow NP VP$	1.0
$NP \rightarrow Det N$	0.2
$NP \rightarrow N$	0.8
$N \rightarrow Adj N$	0.3
$N \rightarrow N$	0.7
$VP \rightarrow VP Adv$	0.4
$VP \rightarrow V$	0.6
$N \rightarrow ideas$	0.1
$N \rightarrow people$	0.2
	$\vdots$
$V \rightarrow sleep$	0.3
	$\vdots$

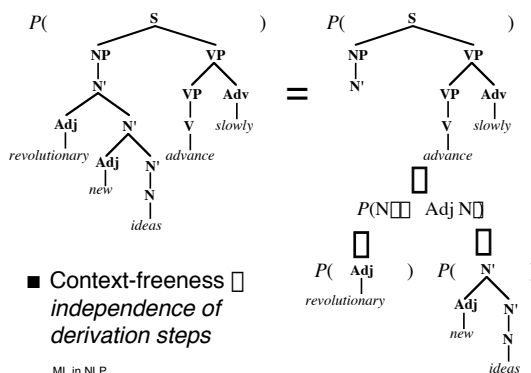
ML in NLP

## Stochastic CFG Inference

- *Inside-outside algorithm* (Baker 79): find rule probabilities that locally maximize the likelihood of a training corpus (instance of EM)
- *Extended inside-outside algorithm*: use information about training corpus phrase structure to guide rule probability reestimation
  - Better modeling of phrase structure
  - Improved convergence
  - Improved computational complexity

ML in NLP

## SCFG Derivations

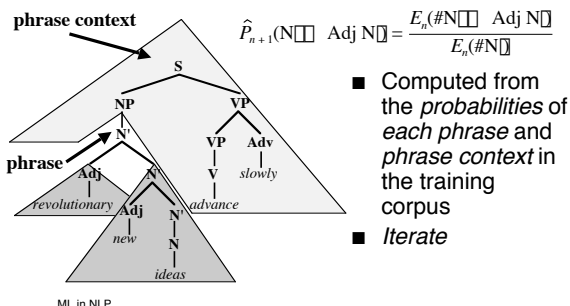


- Context-freeness  $\square$   
*independence of derivation steps*

ML in NLP

## Inside-Outside Reestimation

- Ratios of expected rule frequencies to expected phrase frequencies



## Problems with I-O Reestimation

- Hill-climbing procedure: *sensitivity to initial rule probabilities*
- Does not learn grammar structure directly: only implicit in rule probabilities
- *Linguistically inadequate grammars*: high mutual information sequences are grouped into phrases

((What (((is the) cheapest) fare))((I can) (get ?))))))

Contrast: Is \$300 the cheapest fare?

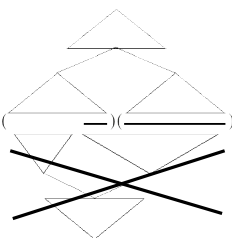
ML in NLP

## (Partially) Bracketed Text

- Hand-annotated text with (some) phrase boundaries

((List (the fares (for (flight) (number 891))))))

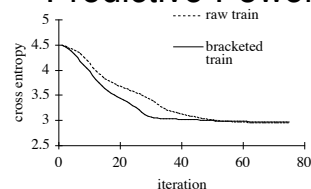
- Use only derivations compatible with training bracketing



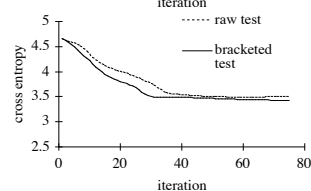
ML in NLP

## Predictive Power

Training:



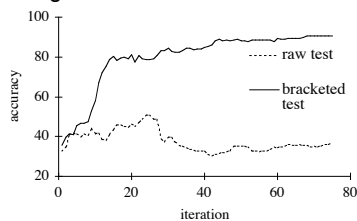
Test:



ML in NLP

## Bracketing Accuracy

- Accuracy criterion: proportion of phrases in most likely analysis compatible with tree bank bracketing



- **Conclusion:** structure is not evident from distribution alone

ML in NLP

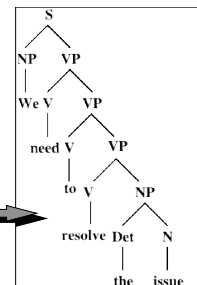
## Limitations of SCFGs

- Likelihoods *independent* of particular words
- *Markovian* assumption on syntactic *categories*

Markov models:

We need to resolve the issue

Hierarchical models:



ML in NLP

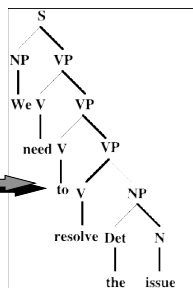
## Lexicalization

- Markov model:

We need to resolve the issue

- Dependency model:

We need to resolve the issue



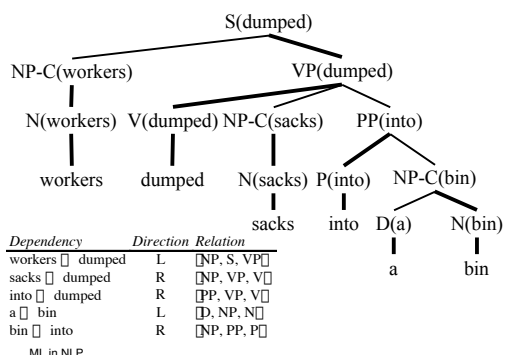
ML in NLP

## Best Current Models

- Representation: surface trees with head-word propagation
- Generative power still context-free
- Model variables: head word, dependency type, argument vs. adjunct, heaviness, slash
- *Main challenge:* smoothing method for unseen dependencies
- Learned from hand-parsed text (treebank)
- Around 90% constituent accuracy

ML in NLP

## Lexicalized Tree (Collins 98)



## Inducing Representations

ML in NLP

## Unsupervised Learning

- Latent variable models
  - Model observables from latent variables
  - Search for “good” set of latent variables
- Information bottleneck
  - Find efficient compression of some observables
  - ... preserving the information about other observables

ML in NLP

## Do Induced Classes Help?

- Generalization
  - Better statistics for coarser events
- Dimensionality reduction
  - Smaller models
  - Improved classification accuracy

ML in NLP

## Chomsky's Challenge to Empiricism

- (1) *Colorless green ideas sleep furiously.*  
 (2) *Furiously sleep ideas green colorless.*  
 ... It is fair to assume that neither sentence (1) nor (2) (nor indeed any part of these sentences) has ever occurred in an English discourse. Hence, in any statistical model for grammaticality, these sentences will be ruled out on identical grounds as equally 'remote' from English. Yet (1), though nonsensical, is grammatical, while (2) is not.

Chomsky 57

ML in NLP

## Complex Events

- What Chomsky was talking about: *Markov models* — state is just a record of observations
- But statistical models can have *hidden state*:
  - representation of past experience
  - uncertainty about correct grammar
  - uncertainty about correct interpretation of experience: *ambiguity*
- Probabilistic relationships involving hidden variables can be induced from observable data alone: *EM algorithm*

ML in NLP

## In "Any Model"?

- *Factored bigram model*:

$$P(w_{i+1} | w_i) \approx \prod_{c=1}^{16} P(w_{i+1} | c) \approx P(c | w_i)$$

$$P(w_1 \dots w_n) \approx P(w_1) \prod_{i=2}^n P(w_i | w_{i-1})$$

$$\frac{P(\text{colorless green ideas sleep furiously})}{P(\text{furiously sleep ideas green colorless})} \approx 2 \times 10^5$$

- Trained for large-vocabulary speech recognition from newswire text by EM

ML in NLP

## Distributional Clustering

- Automatic grouping of words *according to the contexts in which they appear*
- *Approach to data sparseness*: approximate the distribution of a relatively rare event (word) by the collective distribution of similar events (cluster)
- Sense ambiguity  $\approx$  membership in several "soft" clusters
- *Case study*: cluster nouns according to the verbs that take them as direct objects

ML in NLP

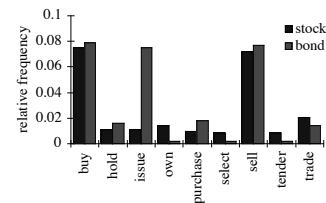
## Training Data

- *Universe*: two word classes  $V$  and  $N$ , a single relation between them (eg. main verb – head noun of verb's direct object)
- *Data*: frequencies  $f_{vn}$  of  $(v,n)$  pairs extracted from text by parsing or pattern matching

ML in NLP

## Distributional Representation

Describing  $n \in N$ : use conditional distribution  $p(V | n)$



ML in NLP

## Reminder: Bottleneck Model

- Markov condition:

$$p(\tilde{n} | v) = \prod_n p(\tilde{n} | n) p(n | v)$$

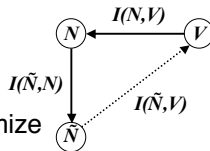
- Find  $p(\tilde{N} | N)$  to maximize mutual information for fixed  $I(\tilde{N}, N)$

$$I(\tilde{N}, V) = \sum_{\tilde{n}, v} p(\tilde{n}, v) \log \frac{p(\tilde{n}, v)}{p(\tilde{n})p(v)}$$

- Solution:

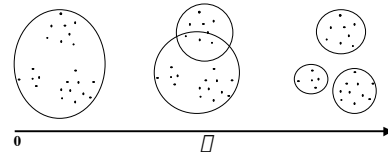
$$p(\tilde{n} | n) = \frac{p(\tilde{n})}{Z_n} \exp(-\beta D_{KL}(p(V | n) \| p(V | \tilde{n})))$$

ML in NLP



## Search for Cluster Solutions

- The scale parameter  $\beta$  ("inverse temperature") determines how much a noun contributes to nearby centroids

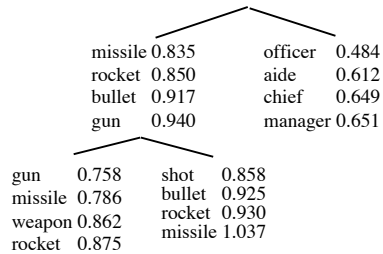


- $\beta$  increases  $\Rightarrow$  clusters split  $\Rightarrow$  hierarchical clustering

ML in NLP

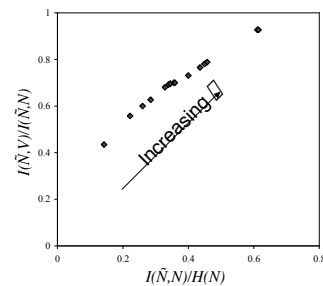
### Small Example

- Cluster the 64 most common direct objects of *fire* in 1988 Associated Press newswire



ML in NLP

### Mutual Information Ratios



ML in NLP

### Using Clusters for Prediction

- Model verb-object associations through object clusters:

$$\hat{p}(v|n) = \frac{1}{\bar{n}} \sum_{\tilde{n}} p(v|\tilde{n}) p(\tilde{n}|n) \quad \text{used in experiments}$$

$$\hat{p}(v|n) = \frac{1}{\bar{n}} \sum_{\tilde{n}} p(v|\tilde{n}) p(\tilde{n}|n) \quad \text{more appropriate}$$

- Depends on  $\bar{n}$
- Intuition*: the associations of a word are a mixture of the associations of the sense classes to which the word belongs

ML in NLP

### Evaluation

- Relative entropy* of held-out data to asymmetric model
- Decision task*: which of two verbs is more likely to take a noun as direct object, estimated from the model for training data in which the pairs relating the noun to one of the verbs have been deleted

ML in NLP

